**Описание работы системы Symphony.Next**

Оглавление

[Требования к установленному программному обеспечению. 1](#_Toc447529235)

[Процедура запуска системы 2](#_Toc447529236)

[Первоначальное конфигурирование системы и проверка условий запуска 2](#_Toc447529237)

[Создание подключения к базе данных 2](#_Toc447529238)

[Чтение настроек для текущего пользователя 3](#_Toc447529239)

[Чтение данных о пользователе и его задачах 3](#_Toc447529240)

[Чтение персональных параметров конфигурирования системы 4](#_Toc447529241)

[Построение пользовательского интерфейса 4](#_Toc447529242)

[Обработка списка автозапуска системы 4](#_Toc447529243)

[Менеджер подключений к базе данных 5](#_Toc447529244)

[Менеджер настроек системы 6](#_Toc447529245)

[Менеджер пользовательского интерфейса 7](#_Toc447529246)

[Запуск задачи 9](#_Toc447529247)

[Закрытие задачи 10](#_Toc447529248)

[Управление плагинами 10](#_Toc447529249)

[Запуск плагина 11](#_Toc447529250)

[Закрытие плагина 11](#_Toc447529251)

[Выполнение акции плагина 12](#_Toc447529252)

## Требования к установленному программному обеспечению.

Система Symphony.Next (далее Симфония) должна располагаться в папке со строго определенной структурой подкаталогов и расположением файлов. Структура каталогов описана в документе «Структура каталога.mmap»:

Симфония располагается в локальном каталоге или сетевом каталоге, который называется корневой каталог системы. В корневом каталоге находится программа Symphony.Next.exe и ряд подкаталогов:

1. Каталоги bin и bin64 – системные каталоги для 32 и 64-х битных версий системы
2. Каталоги plugin и plugin64 – каталоги с 32 и 64-х битными плагинами
3. Каталог Help – каталог, в котором располагаются справочные материалы по системе
4. Каталог Reports – каталог с шаблонами и скриптами отчётов и печатных форм документов
5. Каталог Update – каталог, в котором располагаются файлы обновлений системы.

Расположение тех или иных файлов будет описано далее. Для каталогов bin и plugin подразумевается 32-х разрядная версия системы, и что аналогичные файлы или действия происходят в каталогах bin64 и plugin64 для 64-х разрядной версии системы.

## Процедура запуска системы

Процедура запуска системы состоит из следующих этапов:

1. Первоначальное конфигурирование системы и проверка условий запуска
2. Создание подключения к базе данных
3. Чтение настроек для текущего пользователя (назначенных задач, персональных параметров системы и т.д.)
4. Построение пользовательского интерфейса
5. Обработка списка автозапуска системы.

### Первоначальное конфигурирование системы и проверка условий запуска

За первоначальное конфигурирование системы и проверки условий запуска отвечает программа Symphony.Next.exe, которая находится в корневом каталоге системы.

Проект: Symphony.Next.dproj

Git – репозиторий: /repo/Unicom/Symphony.Next/Symphony.Next.git

Программа в качестве параметра командной строки получает настроечный xml-файл с расширением .csf. Если файла не существует или в командной строке файл не передан, то программа самостоятельно пытается определить конфигурацию системы.

В конфигурацию системы входит следующая информация:

1. Тип и разрядность операционной системы – параметры могут быть определены автоматически по работающей операционной системы и наличия файлов SymDB\*.bpl и SymTask\*.bpl в одном из системных каталогов системы.
2. Тип, сервер и имя базы данных – тип базы данных может быть определен автоматически исходя из наличия файлов SymDB\*.bpl и SymTask\*.bpl в системном каталоге системы.
3. Список автозагрузки – перечень плагинов автоматически запускаемых при загрузке системы. Автоматически не определяется.

Исходя из полученных параметров системы, программа Symphony.Next.exe:

1. выбирает системный каталог;
2. проверяет наличие в нем файлов SymMng.exe, SymDB\*.bpl и SymTask\*.bpl;
3. формирует параметры командной строки для запуска основного приложения системы
4. Запускает программу SymMng.exe из выбранного каталога, передавая ему в качестве параметров командной строки конфигурацию системы:

SymMng.exe –dbt:ТипБД –s:СерверБД –db:ИмяБД –a:plugin1,plugin2,…,plugin

После запуска SymMng.exe, программа Symphony.Next.exe завершает свою работу. Все дальнейшие шаги по запуску системы производятся программой SymMng.exe (далее: программа), исходя из переданных в командной строке параметров.

### Создание подключения к базе данных

Программа: SymMng.exe Проект: SymMng.dproj

Git – репозиторий: /repo/Unicom/Symphony.Next/SymMng.git

Процедура создания подключения к базе данных состоит из следующих шагов:

1. Программа создает менеджер подключений к базе данных – объект, контролирующий загрузку необходимых библиотек и предоставление необходимого соединения по требованию (см. раздел «Менеджер подключений к базе данных»).
2. Программа по параметрам командной строки, определяет тип базы данных, имя сервера базы данных и имя базы данных. Имя сервера и имя базы данных могут быть не указаны, в случае, если диалог авторизации в базе данных позволяет указать требуемые параметры пользователю в ручную.
3. Менеджер подключений создает по требованию программы соединение с базой данных, используя полученные параметры, и устанавливает это соединение как базовое. Базовым соединением становится первое созданное подключение и используется для дальнейшего конфигурирования системы.
4. В случае, если программе не удалось создать подключение к базе данных (отсутствует или поврежден пакет, ошибка авторизации пользователя и т.д.), то программа отображает сообщение об ошибке и завершает свою работу.

### Чтение настроек для текущего пользователя

Чтение настроек для текущего пользователя состоит из двух этапов:

1. Чтение данных о пользователе и зарегистрированных для него в системе задачах
2. Чтение персональных параметров конфигурирования системы

#### Чтение данных о пользователе и его задачах

Чтением данных о пользователе и его задачах занимается пакет SymTask\*.bpl, созданная под конкретный тип базы данных. На текущий момент создан пакет SymTaskORA.bpl, который читает данные из базы данных Oracle.

Проект: SymTaskOra.dproj

Git – репозиторий: /repo/Unicom/Symphony.Next/SymTaskOra.git

Программа SymMng.exe ищет в каталоге, из которого она запустилась, пакет SymTaskТипБД.bpl. Если пакет найден, то он загружается, в нем ищется экспортируемая функция GetTasks и если функция найдена, то она запускается.

Функция GetTasks должна иметь следующее определение:

**function** GetTasks(Session: TObject; UserName: **String**): IXMLNode; **export** ;

Функция принимает в качестве параметра соединение с базой данных (базовое соединение менеджера соединений) и имя текущего пользователя. Функция возвращает XML-данные с двумя блоками: информацией о текущем пользователе и зарегистрированные в системе для пользователя задачи.

После того как функция вернула XML-данные, программа выгружает пакет из памяти.

Образец XML-данных можно посмотреть в файле «tasks.xml». Они состоят из двух разделов: Данные о пользователе (тег USER) и данные о задачах (тег TASKS). Наименования тегов соответствуют наименованиям полей таблиц ssotrudnik, sdivision и streetask.

#### Чтение персональных параметров конфигурирования системы

Чтение и управление персональными параметрами системы осуществляется менеджером настроек - объектом класса TSymMngTuner, который позволяет читать и записывать параметры системы в выбранную базу данных и локальный настроечный файл. Источники данных определяются исходя из выбранного типа базы данных, базового подключения к базе данных и имени пользователя. Подробное описание работы менеджера настроек смотри в разделе «Менеджер настроек системы».

Перечень параметров загружаемых в менеджер настроек зависит от загруженных плагинов, а так же от автора плагина. Для главного приложения системы SymMng.exe менеджер настроек считывает следующие параметры:

1. Имя пакета, который будет создавать и управлять пользовательским интерфейсом списка зарегистрированных для пользователя задач.
2. Имя пакета, который будет создавать и управлять пользовательским интерфейсом акций, загруженных для плагина, а так же панелей инструментов создаваемых плагинов фреймов.

### Построение пользовательского интерфейса

За построение пользовательского интерфейса в программе SymMng.exe отвечает менеджер пользовательского интерфейса – объект класса TUIPackageManager, который инициализируется именами двух пакетов, отвечающих за построение пользовательского интерфейса списка задач и пользовательского интерфейса загруженного плагина. Имена пакетов берутся из настроек пользователя (см. Чтение персональных параметров конфигурирования системы)

После инициализации менеджера пользовательского интерфейса (загрузки выбранных пакетов в память), по запросу программы, пакет, отвечающий за интерфейс списка задач, строит его, опираясь на XML-данные, полученные ранее (см. Чтение данных о пользователе и его задачах).

Подробное описание работы объекта класса TUIPackageManager смотри в разделе «Менеджер пользовательского интерфейса»

### Обработка списка автозапуска системы

Список автозагрузки передается программе в параметре командной строки «а». Список автозагрузки представляет собой имена плагинов, разделенные запятой. Длина списка не ограничена. Допускается указание только имени плагина, без расширения .bpl

Плагины из списка запускаются поочередно стандартным способом, описанном в разделе «Запуск плагина», в порядке указанном в параметре командной строки.

После обработки списка автозагрузки, программа завершила работы по запуску системы и начинает ожидать событий от действий пользователя.

## Менеджер подключений к базе данных

Менеджер подключений к базе данных (далее менеджер подключений) является объектом класса TConnectionMng, объявленного в модуле SymMng.ConnectionMng проекта SymMng.dproj.

Менеджер подключений создается главным окном приложения при старте системы и уничтожается при закрытии главного окна. Менеджер подключений предназначен для создания нового подключения, предоставления уже существующего подключения и в случае, если подключение более не востребовано, удаление подключения.

Менеджер подключения является списком объектов класса TConnection, который определен в том же модуле.

Объект класса TConnection является одним подключением к базе данных. Конструктору объекта TConnection передаются строковые параметры:

* ADBType – тип базы данных (обязательный параметр);
* AServer – имя сервера базы данных (не обязательный параметр);
* ADatabase – имя базы данных (не обязательный параметр).

При создании объекта, соединения с базой данных не происходит, объект сохраняет параметры соединения. При вызове метода Connect, объект создает соединение, выполняя следующие шаги:

1. По типу базы данных в каталоге, откуда была запущена программа, ищется пакет SymDBТипБД.bpl. Например, если требуется соединение с базой данных типа «ORA», то объект будет искать рядом с программой пакет SymDBORA.bpl.
2. Если пакет найден, то он загружается в память, указатель на пакет сохраняется в свойстве Handle объекта.
3. В случае успешной загрузки пакета, в нем ищется и запускается экспортируемая функция  
     
   **function** Connect(**var** Server: **String**; **var** Database: **String**; **var** UserName: **String**): TObject;  
     
   Функции передаются имена сервера, имя базы данных, а затем и возвращаются фактические данные соединения: имя сервера, имя базы данных и имя пользователя через параметры функции, а функция возвращает объект, тип которого определяется разработчиком в пакете SymDB\*.bpl и используемой библиотеки. Объект, возвращаемый функцией Connect, осуществляет фактическое соединение с базой данных. Значения, возвращаемые функцией Connect сохраняются в свойствах объекта.
4. Методу объекта Connect передается строка с именем плагина, для которого создается соединение. При успешном соединении, имя плагина заносится в список подключенных плагинов (свойство Plugins: TStrings). При повторном соединении, объект проверяет этот список, и если плагин уже подключен, то новое соединение не создается, а возвращается объект, сохраненный в свойстве объекта Session.

Объект так же имеет два метода: ReConnect - восстановление соединения с базой данных с помощью загруженного пакета и CloseSession – закрытие соединения для плагина. Плагин, для которого закрывается соединение, удаляется из списка плагинов, а если после удаления плагина список плагинов оказался пуст, то соединение с базой данных разрывается, а пакет выгружается из памяти.

Менеджер подключений содержит два метода

1. Функция GetConnection, ищет в списке соединений имеющееся соединение для указанного типа базы данных имени сервера и базы данных. Если соединение уже создано, то оно возвращается функцией. Если соединения нет, то оно создается и возвращается вызывающему методу.
2. Процедура CloseSession, которая ищет соединение, созданное для указанного плагина, и если оно находится, то соединение для плагина разрывается.

Единственное свойство объекта BaseConnection возвращает базовое соединение, которое было создано при запуск программы и используется для конфигурирования системы.

## Менеджер настроек системы

Менеджер настроек системы является объектом класса TSymMngTuner, объявленного в модуле SymMng.Tuner проекта SymMng.dproj и предназначен для управления списком параметров системы и загружаемых плагинов для каждого пользователя.

Менеджер настроек содержит список групп параметров (интерфейсы типа ISymphonyPlugInCFGGroup), которые в свою очередь являются списком параметров (интерфейсов типа ISymphonyPlugInParam). Оба интерфейса объявлены в модуле SymphonyPlugIn.ParamInterface пакета smplIntf и реализованы в модуле SymphonyPlugIn.ParamImpl пакета smplBaseFrame.

Каждая группа параметров имеет имя, которое совпадает с именем плагина, для которого создана группа или именем приложения. Группа добавляется в менеджер настроек при старте системы или загрузке плагина с помощью метода AddGroup. Данный метод проверяет наличие группы с таким именем в списке загруженных групп, и если группа отсутствует, она создается. Далее группа загружается данными.

Загрузка параметров в группы осуществляется с помощью массива загрузчиков. Загрузчик представляет собой интерфейс типа IConstIO, который объявлен в модуле SymphonyPlugIn.ParamInterface пакета cioIntf.bpl. Менеджер настроек выполняет загрузку параметров с помощью каждого загрузчика, причем каждый следующий загрузчик переписывает значения параметров уже имеющихся в группе и добавляет значения, отсутствующих параметров в группе. В текущей версии менеджера список загрузчиков содержит два загрузчика: сначала загружаются параметры из базы данных, а потом параметры переписываются из локального файла настройки, который создается под каждого пользователя.

Реализация загрузчиков (интерфейсов IConstIO) выполняется с помощью загружаемых пакетов и на текущий момент делится на два типа:

1. Файловый загрузчик – пакет cioXML.bpl – загрузчик считывающий и записывающий параметры в локальный xml-файл. Пакет файлового загрузчика должен содержать экспортируемый метод, с помощью которого программа получает интерфейс загрузчика:  
   **function** GetConstIO(AFileName: **String**): IConstIO ; **export** ;
2. Загрузчик БД – пакет должен иметь имя cioТипБД.bpl. Для типа базы данных «ORA» реализован пакет cioORA.bpl, который считывает и записывает параметры в таблицу sConst базы данных Oracle. Пакет реализующий данный тип загрузчиков должен содержать экспортируемый метод, с помощью которого программа получает интерфейс загрузчика:  
     
   **function** GetConstIO(ASession: TObject): IConstIO ; **export** ;  
     
   Методу передаётся сессия, которая была получена менеджером подключений для заданного типа базы данных.

При запуске системы, главное окно приложения создает объект типа TSymMngTuner и инициализирует список загрузчиков пакетом cioXML.bpl и пакетом для типа базы данных соответствующего базовому подключению. После инициализации загрузчиков, программа создает в менеджере настроек группу с наименованием «SymMng.exe» и загружает её данные, сначала из базы данных, а затем переписывает их значениями из локального xml-файла настроек пользователя.

При загрузке каждого плагина, в менеджере настроек создается группа с именем плагина и данные загружаются в группу сначала из базы данных, затем из файла.

Доступ к значениям параметров осуществляется из менеджера настроек с помощью ряда свойств: ParamValue, AsBoolean, AsDateTime, AsFloat, AsInteger, AsString. Свойства индексированы по имени параметра. Для исключения ошибок, связанных с одноименными параметрами в различных группах, допускается указывать имя параметра с именем группы, разделенными точкой: «MyPlugin.ParamName»

Менеджер настроек содержит метод Edit, позволяющий вызвать диалог редактирования параметров загруженных групп. При вызове группы на диалоговом окне размещается фрейм с параметрами системы (группа с именем «SymMng.exe»), а затем для каждого загруженного плагина проверяется значение возвращаемого функцией TunerFrameClassName интерфейса IPlugInPackage. Если значение имени класса фрейма настройщика определено, то менеджер создает фрейм указанного класса, размещает его на диалоге настройки и инициирует его загрузку. Логика работы фрейма, загрузки и выгрузки определяется автором плагина. Если плагин не требует настроек и фрейм настройщика не создан, то функция TunerFrameClassName интерфейса IPlugInPackage должна вернуть пустую строку.

## Менеджер пользовательского интерфейса

Менеджер пользовательского интерфейса – объект класса TUIPackageManager, объявленного в модуле SymMng.UIPackageManager проекта SymMng.dproj, предназначен для управления пакетами, которые строят пользовательский интерфейс главного окна.

Класс TUIPackageManager является списком объектов класса TUIPackage, который так же объявлен в текущем модуле. В список пакетов пользовательских интерфейсов можно добавлять любое количество элементов, но менеджер будет работать только с пакетами, которые инициированы с помощью метода InitPackages класса TUIPackageManager. Данные пакеты доступны через свойства ActionPackage и TaskPackage. Первый пакет управляет пользовательским интерфейсом загружаемых плагинов, а второй пакет управляет пользовательским интерфейсом списка задач. Имена пакетов задаются в настройках пользователя или выбираются автоматически и числа доступных пакетов, удовлетворяющих требованиям, предъявляемым к пакетам управления пользовательского интерфейса:

1. Пакет должен находиться в той же папке, откуда запускается программа SymMng.exe
2. Имя пакета должно соответствовать формату stb\*.bpl.
3. Пакет должен экспортировать две функции: PackageDesc и GetUIManager

Функция PackageDesc возвращает XML-данные с описанием пакета.

**function** PackageDesc: IXMLNode ; **export** ;

XML-данные содержат следующие теги:

* CAPTION – заголовок пакета. Заголовок будет отображаться в окне настроек системы при выборе варианта построения интерфейса
* HINT – краткое описание пакета
* PARENTCLASS – имя класса объекта, который необходим пакету для построения пользовательского интерфейса.

Функция GetUIManager возвращает интерфейс, с помощью которого менеджер пользовательского интерфейса управляет работой пакета:

**function** GetUIManager: ISymphonyUIManager ; **export** ;

Интерфейс ISymphonyUIManager объявлен в модуле stbIntf.MainUnit пакета stbIntf.bpl

Интерфейсом ISymphonyUIManager объявлены следующие методы:

**function** BuildTasks(Parent: TComponent; Tasks: IXMLNode;   
 Events: TEventList): Boolean ;

Метод создает пользовательский интерфейс для списка задач, который ему передан в параметре Tasks (см. раздел «Чтение данных о пользователе и его задачах»). Через параметр Parent передается компонент, на котором пакет будет строить пользовательский интерфейс. Тип компонента должен соответствовать параметру PARENTCLASS, возвращаемого функцией PackageDesc пакета. Если параметр не задан, то пакет должен самостоятельно его создать или выбрать из уже размещенных на форме компонентов. Через параметр Events передается перечень указателей на обработчики предопределенных событий (старт задачи, запуск акции, закрытие задачи, печать и т.д.)

**function** TaskForControl(AControl: TObject): IXMLNode ;

Функция возвращает задачу, для переданного объекта. Метод используется для поиска задачи, которую необходимо запустить при щелчке на элементе управления, который построил данный пакет.

**function** ControlForTask(ATask: IXMLNode): TObject ;

Функция возвращает созданный пакетом элемент управления для заданной в параметре задачи.

**function** BuildActions(Parent: TComponent; Actions:   
 ISymphonyPlugInActionList; Events: TEventList): Boolean ;

Метод создает пользовательский интерфейс для списка акций загружаемого плагина. Список акций передается через параметр Actions, список обработчиков предопределенных событий передается через параметр Events. Через параметр Parent передается компонент, на котором пакет будет строить пользовательский интерфейс. Тип компонента должен соответствовать параметру PARENTCLASS, возвращаемого функцией PackageDesc пакета. Если параметр не задан, то пакет должен самостоятельно его создать или выбрать из уже размещенных на форме компонентов.

**procedure** DestroyActions(Actions: ISymphonyPlugInActionList) ;

Метод удаляет элементы пользовательского интерфейса которым соответствуют акции из переданного через параметр Actions списка акций.

**function** ActionForControl(AControl: TObject): ISymphonyPlugInAction ;

Функция возвращает акцию для заданного параметром AControl элемента управления

**function** ControlForAction(AAction: ISymphonyPlugInAction): TObject ;

Функция возвращает элемент управления привязанного к заданной акции

**procedure** MergeUI(AForm: TForm) ;

**procedure** UnMergeUI(AForm: TForm) ;

Процедуры объединяют и разъединяют пользовательский интерфейс главного окна и дочернего окна, переданного в качестве параметра.

**function** SetContextVisible(AForm: TForm;   
 AVisible: Boolean): boolean ; **overload** ;

**function** SetContextVisible(AAction: ISymphonyPlugInAction;   
 AVisible: Boolean): boolean ; **overload** ;

**function** SetContextVisible(AContextName: String;   
 AVisible: Boolean): boolean ; **overload** ;

Функции показывают или скрывают пользовательский интерфейс, привязанный к одному из параметров функций: форме, акции или группе акций с совпадающим именем контекста.

## Запуск задачи

В начале работы программа загружает список задач для текущего пользователя (см. раздел «Чтение данных о пользователе и его задачах») и строит пользовательский интерфейс для списка задач исходя из текущих настроек пользователя (см. разделы «Чтение персональных параметров конфигурирования системы» и «Построение пользовательского интерфейса»).

При построении пользовательского интерфейса списка задач, для каждой задачи, пакет, отвечающий за пользовательский интерфейс, создает элемент управления (например, кнопку), настраивает его внешний вид и назначает обработчику сообщения активизации элемента (например, щелчка по кнопке) процедуру главного окна OnTaskExecute. Указатель на процедуру главного окна OnTaskExecute передается через параметр Events процедуры BuildTasks интерфейса ISymphonyUIManager, который обязан реализовывать пакет. Тип создаваемого элемента управления, внешний вид и расположение элемента полностью определяется пакетом, который строит пользовательский интерфейс.

Когда пользователь активизирует элемент управления созданного пакетом для задачи (например, щелкает по кнопке), вызывается обработчик события OnTaskExecute. В теле обработчика, программа, через свойство TaskPackage менеджера пользовательского интефейса (см. раздел «Менеджер пользовательского интерфейса»), вызывает метод TaskForControl интерфейса ISymphonyUIManager, который возвращает данные о привязанной задаче.

Программа считывает значение параметра задачи UNIQUE, и если оно равно True, то ищет задачу в списке открытых задач. Если задача найдена, то активизируются все дочерние окна главного окна приложения, которые привязанные к данной задаче.

Если значение параметра UNIQUE равно False или задача не найдена в списке открытых задач, то программа считывает атрибут задачи FILENAME, и если значение атрибута определено, то программа запускает плагин с указанным именем (см. раздел «Запуск плагина»), а затем запускает акции плагина с установленным признаком AutoRun (см. раздел «Выполнение акции»).

Запущенная задача помещается в список открытых задач.

## Закрытие задачи

При построении интерфейса пользователя для открывающегося плагина, пакет, отвечающий за его построение, может создать элемент управления, отвечающий за закрытие плагина и задачи. Для этого, методу построения интерфейса пользователя BuildActions интерфейса ISymphonyUIManager передается указатель на метод главного окна OnClosePlugIn.

При активизации данного элемента управления, главное окно приложения в методе OnClosePlugIn, вызывает метод DestroyActions интерфейса ISymphonyUIManager пакета, доступного через свойство ActionPackage менеджера пользовательского интерфейса, и закрывает плагин (см. раздел «Закрытие плагина»).

Метод DestroyActions интерфейса ISymphonyUIManager, который реализуется пакетом построения пользовательского интерфейса, должен уничтожить все элементы управления, которые были построены для акций, входящих в список акций, переданный методу в качестве параметра.

## Управление плагинами

Плагины системы представляют собой библиотеки - пакеты созданные средствами Embarcadero Delphi. Файлы плагинов должны располагаться в каталогах plugin и plugin64 (в зависимости от разрядности пакета). Пакет плагина должен экспортировать функцию PackageInterface, которая должна вернуть интерфейс IPlugInPackage, с помощью которого программа будет управлять плагином. Интерфейс IPlugInPackage определен в модуле SymphonyPlugIn.PackageInterface, который входит в состав пакета smplIntf.bpl.

На главной форме приложения SymMng.exe располагается компонент PlugInMng класса TPlugInManager (Модуль: SymphonyPlugIn.PlugInManager, Проект: smplmng.dproj, git-репозиторий: /repo/Unicom/Symphony.Next/PlugIn\_Core.git). Компонент предназначен для управления загрузкой и выгрузкой плагинов. Компонент не имеет собственных свойств, но имеет ряд событий, которые обрабатывает главная форма приложения:

* OnGetOwnerForm – данное событие происходит, когда плагин создает фрейм, который нужно разместить на форме. Обработчик данного события должен предоставить компоненту экземпляр формы. Если обработчик события не определен или форма не будет предоставлена, то создание фрейма прерывается.
* OnGetSession – данное событие происходит, когда плагину необходимо соединение с базой данных. Обработчик должен вернуть соединение компоненту исходя из типа базы данных, для которого написан плагин. Отсутствие соединение не прерывает выполнение плагина.
* OnGetParams – в обработчике данного события можно передать дополнительные параметры в момент выполнения. Например, снять требование уникальности фрейма данного класса среди загруженных фреймов.
* OnClosePlugIn – событие возникает, когда плагин закрывается.
* OnLoadActions – событие возникает, когда плагин загружен и он вернул вызывающему приложению список своих акций (см. раздел «Запуск плагина»). Обработчик данного события может выполнить с акциями плагина какие либо операции, например, проверить права доступа и построить интерфейс пользователя для запуска акций плагина.

## Запуск плагина

Запуск плагина происходит по команде пользователя при открытии задачи или при старте системы, когда обрабатывается список автозагрузки. Исходными данными для запуска плагина является его имя.

Запуск плагина осуществляется с помощью метода LoadPlugIn компонента PlugInMng класса TPlugInManager, который расположен на главном окне приложения. Метод проверяет, есть ли в списке загруженных плагинов плагин с таким именем, и если его нет, то создает объект класса TPlugIn и в случае удачной его загрузки, добавляет его к списку открытых плагинов.

Объект класса TPlugIn, по переданному имени плагина и разрядности системы определяет полный путь к файлу плагина. Если файла не существует, то загрузка считается неудачной и она прерывается.

Если файл найден, объект загружает пакет в память, сохраняя в своем свойстве Handle указатель на пакет, а затем ищет экспортируемую функцию PackageInterface, которая должна вернуть интерфейс IPlugInPackage. С помощью полученного интерфейса объект загружает список акций и вызывает обработчик события OnLoadActions, в котором могут быть проверены права доступа и построен интерфейс пользователя (см. раздел «Менеджер пользовательского интерфейса»)

После загрузки интерфейса, плагин считается загруженным, а обработчики запуска задачи или автозагрузки системы запускают акции с установленным признаком AutoRun (см. раздел «Выполнение акции плагина»).

## Закрытие плагина

Закрытие плагина происходит во время закрытия задачи, для которой плагин был запущен. Необходимо реализовать автозакрытие плагина после выполнения разовой работы.

Плагин закрывается при вызове метода ClosePlugIn компонента TPlugInManager, которому в качестве параметра передается объект TPlugIn, или при вызове метода Stop объекта TPlugIn.

При закрытии плагина объект TPlugIn закрывает все дочерние окна главного окна приложения, которые были созданы для него, вызывает обработчик события OnClosePlugIn (если он определен), помечает плагин как остановленный и запускает процедуру уничтожения остановленных плагинов.

Процедура уничтожения остановленных плагинов реализована в виде периодически вызываемого метода компонента TPlugInManager, в которой при первом проходе уничтожаются ссылки на акции плагина, а при втором проходе выгружается пакет из памяти и уничтожается сам объект плагина TPlugIn. Если компонент TPlugInManager не находит ни одного остановленного плагина, то процедура уничтожения останавливается. В промежутках между вызовами функции сначала штатными средствами завершают работу дочерние формы, при втором вызове дается таймаут для сборщика мусора, который удаляет интерфейсы.

Если в любой момент до выгрузки пакета из памяти программа попытается загрузить плагин заново, то компонент TPlugInManager снимает с плагина признак остановленного, восстанавливает при необходимости акции плагина и выполняет акции с признаком AutoStart

## Выполнение акции плагина

Акция плагина представляет собой интерфейс типа ISymphonyPlugInAction, который объявлен в модуле SymphonyPlugIn.ActionInterface проекта smplIntf.dproj (git-репозиторий: /repo/Unicom/Symphony.Next/PlugIn\_Core.git).

Акция плагина – это «точка входа» для плагина, которую может получить основная программа и выполнить. Большинство свойств акции предназначено для построения пользовательского интерфейса. Три свойства акции позволяют описать код, который будет вызван главным приложением:

1. FrameClassName – имя класса фрейма, который будет создан при выполнении акции. Данный класс фрейма, должен содержаться в пакете плагина, наследоваться от класса TSymphonyPlugInBaseFrame или одного из его потомков и экспортировать класс с помощью вызова функции RegisterClass.
2. PlugInMethodName – имя метода, который будет вызван при выполнении акции. Данный метод должен располагаться в пакете плагина и экспортироваться с помощью директивы Export.
3. Command – интерфейс типа ISymphonyPlugInCommand, который объявлен в модуле SymphonyPlugIn.ParamInterface. Данный интерфейс позволяет передавать неограниченное число параметров различных типов для выполняемого в рамках акции кода.

При выполнении акции проверяется сначала значение свойства FrameClassName и если оно задано, то поочередно выполняются следующие шаги:

1. По имени класса фрейма получаем сам класс фрейма
2. Запрашиваем у главного приложения системы и получаем окно для размещения фрейма
3. Создаем фрейм, передавая ему в конструкторе соединение с базой данных и параметры из свойства Command. На данном этапе фрейм должен, используя переданные объекты, подготовиться к работе (настроить компоненты работы с базой данных, настроить интерфейс, инициализировать параметры в соответствии с переданными данными через свойство Command.
4. Фрейм размещается на окне
5. Окно регистрируется в свойстве акции Forms.

Если имя класса фрейма не задано, то программа анализирует значение свойства PlugInMethodName. Если значение свойства задано, то ищется в пакете плагина экспортируемая функция с заданным именем, и в случае если функция найдена, она вызывается. Функции в качестве параметров передается соединение с базой данных, а так же данные, определенные в свойстве Command акции. Функция должна вернуть значение True, если выполнение успешное, и значение False, если в теле функции произошли ошибки.